EXHIBIT 2

From:	Brian Swetland.	Sent:8/9/2005 7:00 AM.
To: [-]	Patrik Reali.	
Cc: [-]	Sascha Brawer; arubin@google.com; nicksears@google.com; miner@google.com; cwhite@google.com; fadden@google.com; tcole@google.com; ficus@google.com; Urs Hoelzle; Robert Griesemer.	
Bcc: [-]	.	
Subject:	Re: Java VM for Android.	

On 8/9/05, Patrik Reali <patrik@google.com> wrote:
> IMHO, another interesting project is JNode [1], a complete JVM running on
> the bare metal with just a tin layer of assembly. AFAIK, it has a linker for
> external files, which may fit your needs. The current version is for x386,
> but the (really nice) design makes it unaware of this and should allow for a
> rather simple retargeting of the whole projects.
>
> JNode shows that you can use java for everything without need of using
> another system or language and still get a good performance, in part because
> the system is simpler and you can avoid loss due to design choices made to
> fit more than one system (e.g running linux + java at the same time)

This sounds very similar to what I built at Danger -- and that
certainly worked incredibly well for the Hiptop. One of our goals
here is to support native (C/C++) code as well as non-java
higher level languages, so I'm less convinced that an "entire world on
top of java" solution is ideal. Linux is desirable because it gives
us a huge head start on driver support, a full tcp/ip stack, etc --
the chipset vendors like TI are supporting linux development and the
OEMs are interested in it.

> If you want to have an open-source java platform, you should really
> consider Classpath, as the project has an enourmous experience in the java
> libraries and regroups all the JVM developers that are done with the JVM
> implementation and are now improving the libraries (not to mention that
> getting the libraries and testsuite right is at least 10 times more work
> than the VM). They currently aim for J2SE but it would be great if you would
> help them to build a J2ME version of the classes.
>
> Classpath is licensed in GPL+Exception, which allows you to ship the
> libraries in a LGPL fashion. Classpath has also the advantage that you can
> use it almost right away (it is very portable) and then decide which pieces
> to optimize (let me guess: String....).

I'm strongly leaning towards pre-linking and caching the prelinked
class files -- I'm not sure what impact this would have on LGPL-like
licensing. I'm actually pretty fuzzy about the LGPL in general for
embedded devices where the system is probably not user servicable
(cell carriers are generally not very keen on end users replacing
kernels, java libraries, etc).

The Hiptop VM saved a ton of space (in flash and over the network),
and startup time by have the libraries (cldc java core, danger
toolkis, midp2) and apps pre-linked so they were pretty much "load and
go" with very minimal linking needed at runtime. Just combining
constant pools in typical sets of classfiles results in a ~50% space
savings in my experience.

It would be interesting to see how heavy the classpath stuff is
(space-wise) "out of the box", once it has been trimmed down to CLDC
or CDC spec levels. Performance definitely can be addressed by
pushing stuff native (though again I am curious what that means from a
license standpoint). I'll definitely look at it for vm bootstrapping
-- having a big pile of java code to start mucking with is always

nice.

Do you know if classpath can be build using jikes? I'm avoiding using
Sun tools, libraries, etc, until such time as we have some agreement
with them regarding being able to build a cleanroom VM and distribute
source without harrassment.

- Brian